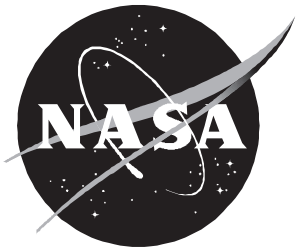
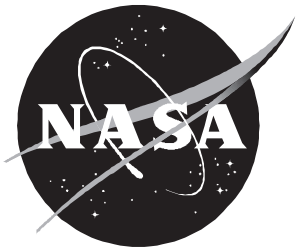


NASA Technical Memorandum 4607

Automatic Procedures for Computing Complete Configuration Geometry From Individual Component Descriptions

Raymond L. Barger and Mary S. Adams

July 1994



NASA Technical Memorandum 4607

Automatic Procedures for Computing Complete Configuration Geometry From Individual Component Descriptions

Raymond L. Barger and Mary S. Adams
Langley Research Center • Hampton, Virginia

National Aeronautics and Space Administration
Langley Research Center • Hampton, Virginia 23681-0001

July 1994

Introduction

The wave-drag format (ref. 1) is a convenient way to describe a rough-cut airplane geometry for making a rapid linear analysis of the configuration aerodynamics. The format is simple and concise, and it allows changes to be made easily. One reason for this simplicity is that the individual configuration components are described as disjoint surfaces. This limitation is of little consequence for linear analysis codes, but nonlinear codes require a complete surface description to establish a computational grid.

The purpose of this paper is to describe automatic procedures for completing the geometry, beginning with a wave-drag geometry description. A method for automatically calculating the wing-fuselage intersection line was presented in reference 2. That method is limited to configurations for which the fuselage has circular cross sections. Here, a procedure is described that computes the intersection of two wing-like surfaces. These two procedures permit one to compute all the intersection lines and thereby complete the geometry for a configuration that includes a canard, a horizontal tail, nacelles, pylons, and a vertical fin on the fuselage or on the wing.

Symbols

$\mathbf{c}$	vector location of generic point on camber line
$c(x)$	z coordinate of camber line as a continuous function of x
$d(t)$	absolute distance (eq. (5))
$e(t)$	error indicating extent to which $\mathbf{v}$ is displaced from fuselage surface
$\hat{\mathbf{i}},\hat{\mathbf{j}},\hat{\mathbf{k}}$	orthonormal base vectors
$\mathbf{r}$	vector location of generic point on fuselage surface
$r(x)$	fuselage radius as a continuous function of x
t	parameter that controls length of $\mathbf{v}$ (eq. (4))
$\mathbf{v}$	variable-length vector defined by equation (4)
v_x, v_y, v_z	scalar components of $\mathbf{v}$
$\mathbf{w}$	position vector of point on upper surface lofting line at airfoil section location
x, y, z	Cartesian coordinates
z_i	interpolated value of z
θ	angular coordinate
θ_b	bounding value of θ

Procedure

Preliminary Considerations

Two procedures are used to compute the intersection lines of adjacent components. The first is the method described in reference 2 which is applicable when one of the surfaces has circular cross sections. The second technique assumes that both surfaces are input as a set of airfoil sections. From a purely mathematical point of view, only the second technique is required because surfaces input in the first (circular cross-section) format could be converted into the equivalent of the second format. However, the coding and computational advantages of the circular cross-section format are significant; consequently, a separate algorithm is retained for that case. The two techniques appear as separate subroutines in the computer codes.

These two techniques are described herein. Then, some problems that can arise in particular cases are described, together with the methods that are used to resolve them. A discussion of output format and surface description considerations concludes the analysis.

Intersection Line When One Component Has Circular Cross Sections

In the wave-drag format, a wing, pylon, canard, or fin is specified by individual airfoil sections. Such components are denoted wing-like surfaces. The individual airfoil sections are prescribed at constant-span y stations for a wing, canard, or horizontal tail, and at constant vertical z stations for a fin or pylon. A line that connects the corresponding point (e.g., the third point) of the various sections is called a lofting line. (See fig. 1.)

If this surface intersects the fuselage, which has circular cross sections, the intersection line is determined by the method of reference 2. For the reader's convenience, that method is briefly reviewed here.

The fuselage is input as a set of circles at specified x stations, centered at points determined by the input fuselage camber line coordinates at these x stations. Since camber ordinates can be interpolated at any x location from the input camber line array, the shape of the camber line can be represented by a function $c(x)$ that is defined by these interpolated values. Similarly, a radius distribution $r(x)$ can be synthesized from the input array of fuselage radii. A

surface equation for the fuselage can then be written as

$$\mathbf{r}(x,\theta) = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}} \tag{1}$$

where

$$y(x,\theta) = r(x)\cos\theta \tag{2}$$

$$z(x,\theta) = c(x) + r(x)\sin\theta \tag{3}$$

The wing lofting lines are extrapolated inward to the fuselage as follows. Beginning on the upper surface, let $\mathbf{w}_1$ be the vector location of a point on the first airfoil section (i.e., the most inboard section), and let $\mathbf{w}_2$ be the location of the corresponding point on the second airfoil section. Thus, $\mathbf{w}_1$ and $\mathbf{w}_2$ lie on the same lofting line. A vector of variable length $\mathbf{v}$ that is collinear with these two points and pointing inward toward the fuselage is

$$\mathbf{v} = \mathbf{w}_1 + t(\mathbf{w}_1 - \mathbf{w}_2) \tag{4}$$

where t is a small but otherwise arbitrary number. If the x coordinate of $\mathbf{v}(t)$ is denoted v_x and the vector location of the camber line at $v_x(t)$ is denoted $\mathbf{c}(t)$, then the distance (fig. 2) from $\mathbf{c}(t)$ to $\mathbf{v}(t)$ is

$$d(t) = |\mathbf{v}(t) - \mathbf{c}(t)| \tag{5}$$

This distance is compared with the fuselage radius $r(v_x)$ at v_x to find the error e. Thus,

$$e(t) = d(t) - r[v_x(t)] \tag{6}$$

Now the value of t in equation (4) is incremented by an amount proportional to $e(t)$, and the procedure is iterated until $e(t)$ is below a specified error bound. The components of $\mathbf{v}$ are then taken as a point on the wing-fuselage intersection curve.

Examples of intersection lines computed in this manner are shown in figures 3–6. Figure 3 shows a wing-fuselage intersection. Figure 4 shows the canard-fuselage intersection line. Figure 5 shows intersections of vertical and horizontal tail surfaces with the fuselage. Figure 6 shows a pylon-nacelle intersection.

Intersection Line When Neither Component Has Circular Cross Sections

The intersection of a vertical fin with the wing (fig. 7) can be taken as a model for this type of problem. The fin section that is closest to the wing is denoted by the subscript 1 and the second section, by 2. A fin lofting line is extrapolated toward the wing surface in accordance with equation (4).

On the wing upper surface, a new wing section is interpolated at $v_y(t)$. On this interpolated section the wing ordinate is denoted $z_i(x)$. At $x = v_x$ this coordinate is compared with v_z to determine the error. Thus,

$$e(t) = v_z - z_i(v_x) \tag{7}$$

This error represents the vertical distance between the extrapolated lofting line and the wing surface. To reduce the error, t is incremented by an amount proportional to $e(t)$, and the entire procedure is repeated. This iteration is continued until the absolute value of the error is below a specified bound. The set of intersection points obtained by extrapolating all the lofting lines to the wing surface defines the fin-wing intersection line. The intersection of a pylon with the wing lower surface is handled in precisely the same manner.

The same basic procedure, with some terminology changes, can also be used to compute the intersection of a wing, canard, or fin with a noncircular fuselage. In this case the fuselage is input as a set of circumferential curves (y and z coordinates) at constant x stations. Now, when a lofting line is extrapolated toward the fuselage in accordance with equation (7), a new circumferential curve $y_i(z)$ is interpolated at $x = v_x$. At $z = v_z$ this coordinate is compared with v_y. Thus, equation (6) is replaced by

$$e(t) = v_y - y(v_z) \tag{8}$$

Figures 7 and 8 show examples of fin-wing intersection lines and pylon-wing intersection lines computed in this manner.

Potential Problems

The above description for computing the intersection of a wing-like surface with a circular cross-section fuselage (eqs. (1)–(6)) does not actually require that the fuselage and the wing-like surface be disjoint—that is, the root airfoil section need not lie entirely outside the fuselage. As a rule the intersection line algorithms converge even if one of the input components actually pierces through the other. For example, the root wing section may lie partly or entirely within the fuselage.

However, one situation that can cause a problem is illustrated in figure 9. Here, a vertical fin intersects the fuselage near the aft end where the fuselage radius is relatively small. The extended fin lofting lines intersect the fuselage in two places: one on the upper half of the fuselage surface and the other on the lower half. The input fin is set so low that, for

some of the lofting lines, the tip of the line is closer to the lower intersection point than is the upper point. Consequently, the intersection line algorithm will converge to the lower point. Therefore, to assure that the correct intersection point is computed, the vertical fin is required to be input as a surface that is disjoint from the fuselage, and a test has been written into the program to provide a check on this. This problem cannot arise if the fin is set on the upper wing surface because the upper surface is labeled separately from the lower surface.

Another problem that can arise is that the lofting lines of a surface may not all intersect the second surface. Thus, mathematically, $\mathbf{v}$ in equation (4) does not intersect the second surface for any value of t. As presently constituted, the algorithms do not make allowance for this situation. However, a procedure is described in reference 2 for treating the case in which the wing is set so low on the fuselage that some of the lower surface lofting lines do not intersect the fuselage.

Output Considerations

The output geometry can be expressed in a Hess format. (See ref. 3.) All the intersection lines are computed before any output is printed.

Fuselage. In the output the fuselage is described as separate upper and lower surfaces. For circular cross sections, the defining points are computed from polar coordinates, which are used in each cross section (x = Constant) with the polar axes centered at $y = 0$ and $z = c(x)$. The angular coordinate $\theta = 0$ corresponds to the horizontal axis $z = c$. The fuselage upper surface is distinguished from the lower surface by a bounding value $\theta_b(x)$, which is defined as follows. From the nose to the leading edge of the canard, $\theta_b(x)$ varies continuously from 0 to the value of $\theta_b(x)$ that corresponds to its value at the intersection of the canard leading edge with the fuselage. In the canard region, $\theta_b(x)$ corresponds to the intersection line of the canard upper surface and the fuselage. Between the canard trailing edge and the wing, $\theta_b(x)$ varies from its value at the canard trailing edge to that at the wing leading edge intersection point. In the wing region it corresponds to the wing upper surface intersection line. A similar procedure is followed aftward from the wing trailing edge to the horizontal tail, over the tail, and to the end of the fuselage, where $\theta_b(x)$ terminates with a value of 0. The fuselage lower surface is bounded by a curve that is identical to $\theta_b(x)$, except in the canard, wing, and tail regions, where it corresponds to the lower surface intersection lines.

If the fuselage cross sections are not circular, the y and z coordinates of each cross-section curve are expressed parametrically in terms of arc length. Then, the bounding quantity $\theta_b(x)$ is replaced by a similarly defined bounding value of the arc length parameter.

For continuity the same number of points is printed out for each cross section. However, the number of points is not evenly divided between upper and lower surfaces. The fraction of the points that are assigned to the upper surface is chosen so that the points will be approximately evenly spaced at the x location where the wing leading edge intersects the fuselage. This system is set arbitrarily and could easily be replaced with any other criterion.

At any cross section, upper surface points are spaced at equal distances, beginning at $\theta = \pi/2$ and ending at $\theta = \theta_b$, except in the region of a vertical fin where the fuselage points begin at the value of θ that corresponds to the fin-fuselage intersection point at that station. On the lower surface the points are spaced at equal distances, beginning at $\theta = -\pi/2$ and terminating at the lower surface bounding curve.

Wings. The wing, canard, and horizontal tail components are treated as having separate upper and lower surfaces. In the output the airfoil sections are listed in order starting at the fuselage. The input root section is replaced by the intersection line with the fuselage because the input root section might lie partly or totally within the fuselage.

If a vertical fin is set on the wing upper surface, two additional curves must be interpolated into the wing output coordinates. The first curve begins at the wing leading edge at $y = y_f$, where the fin leading edge intersects the wing. It follows the wing section shape back to the fin leading edge, then traces the intersection line of the inner fin surface with the wing to the fin trailing edge, then follows the wing surface along $y = y_f$ to the wing trailing edge. The second curve lies identically on the first ahead of the fin and behind it but traces the intersection line of the fin outer surface with the wing in the fin region. In the wing lower surface output, a surface curve is interpolated at $y = y_f$ to assure continuity of the wing construction lines.

On the wing lower surface, the intersections of the pylons are treated in a manner similar to the fin-wing intersections on the upper surface. For each pylon-wing intersection, two curves are interpolated on the lower surface (one tracing the inner part of the intersection line and the other tracing the outer part), and one curve is interpolated on the upper surface.

Empennage. The canard, fin, and horizontal tail surfaces are specified in the wave-drag format by the root and tip sections only. However, for present purposes, several airfoil sections are interpolated between the root and tip. Then, in the output file, the root section is replaced by the intersection line, as discussed earlier.

Pylons and nacelles. The input format requires that the nacelles be described as bodies of revolution. For the output, the cross sections are circles except in the region of the pylon-nacelle intersection, where the circular arc begins at the inner intersection line and ends at the outer intersection line. For continuity, cross sections are interpolated at x stations that correspond to the pylon-nacelle intersection points.

The pylons present a special problem. In the input file they are specified by two airfoil sections (as fins). The nacelles are normally set close to the wing; therefore, the input upper pylon section may pierce through the wing, and the lower pylon section may pierce the nacelle. This event is even more likely if the geometry is being automatically modified, as in an optimizer loop. Therefore, the two input pylon sections are used only for the purpose of computing the intersection lines. For the output, the upper section is replaced by the pylon-wing intersection line, and the lower section is replaced by the pylon-nacelle intersection line. Of course, even this precaution will fail if the input geometry is such that the nacelle pierces the wing.

Continuity. The object of this analysis is to define a complete configuration surface geometry, starting from disjoint component geometries in wave-drag format. The goal is not to design a configuration or to generate a grid, but to provide a tool for those purposes. Nevertheless, to facilitate graphical display and surface grid generation, an effort was made to preserve continuity of output surface lines where feasible. For example, an additional fuselage curve that connects to each point of the canard-fuselage intersection line is relatively easy to obtain by interpolation. Consider, however, the problem that arises when both fin and horizontal tail surfaces are present. In general, they will not begin and end at the same x stations but will have a region of overlap in the x intervals. (See fig. 5.) In this overlap region, not only must the fuselage lines be interpolated but also an additional fin lofting line that corresponds to each tail lofting line and vice versa. The accounting problem is even more severe on wings with nacelles. For example, each pylon lofting line would have to be continued along the wing, the other pylon, both nacelles, the fuselage, the wing upper surface, and the wing tip. Constructing all these interpolated curves would succeed in closing the tiny gaps that occur in graphical displays of surfaces when the construction lines are not continuous. But that construction would not produce a useful surface grid for computational purposes because the curves are erratically distributed. However, a more smoothly distributed set of grid lines could be interpolated from the surface construction lines.

Some examples of complete configuration geometries computed by the methods described herein are shown in figures 10–12.

Concluding Remarks

Procedures have been derived for developing a complete airplane surface geometry starting from component descriptions. The procedures involve locating the intersection lines of adjacent components and omitting any regions for which part of one surface lies within the other. Two algorithms were used: one, if both of the intersecting surfaces are wing-like surfaces; the other, if one of the surfaces has circular cross sections. Some sample results in graphical form were included.

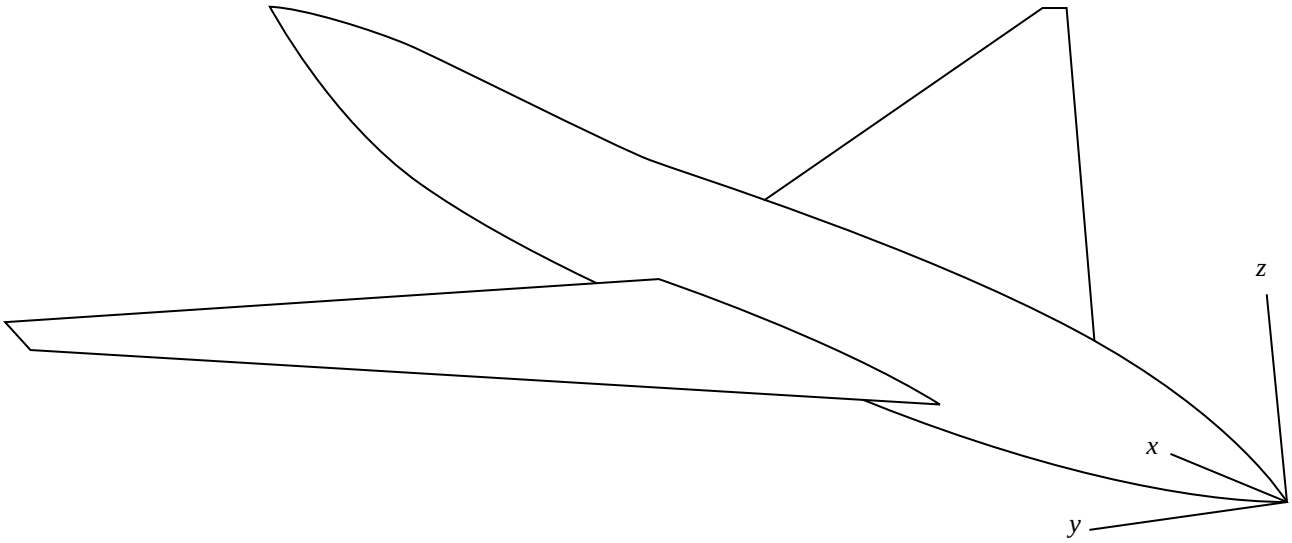
NASA Langley Research Center
Hampton, VA 23681-0001
May 24, 1994

References

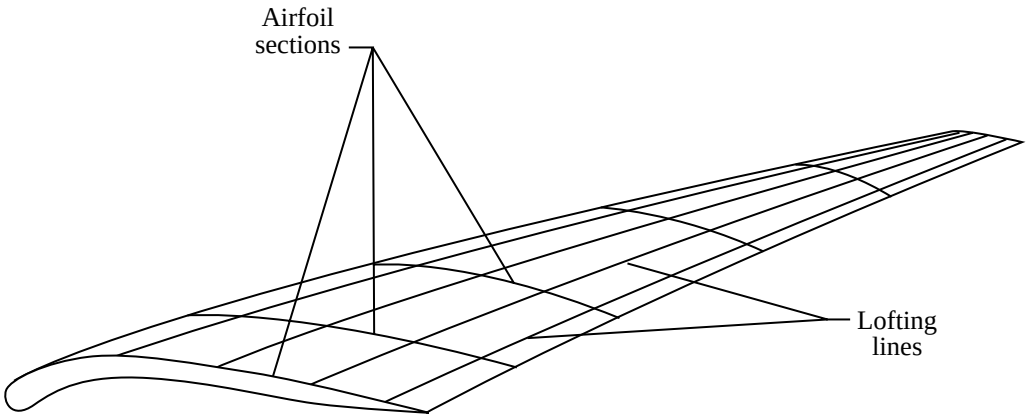
1. Craidon, Charlotte B.: *User's Guide for a Computer Program for Calculating the Zero-Lift Wave Drag of Complex Aircraft Configurations.* NASA TM-85670, 1983.

2. Barger, Raymond L.; and Adams, Mary S.: *Automatic Computation of Wing-Fuselage Intersection Lines and Fillet Inserts With Fixed-Area Constraint.* NASA TM-4406, 1993.

3. Halsey, N. D.; and Hess, J. L.: *A Geometry Package for Generation of Input Data for a Three-Dimensional Potential-Flow Program.* NASA CR-2962, 1978.



(a) Coordinate system.



(b) Wing geometry.

Figure 1. Coordinate system and basic wing geometry.

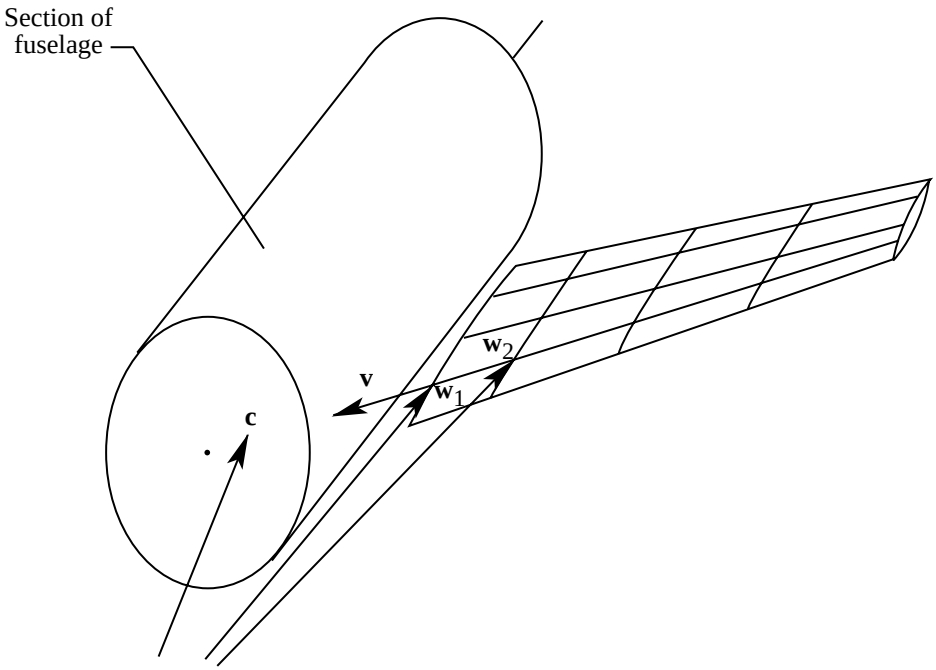


Figure 2. Extrapolation of wing lofting line toward fuselage.

(a) Original configuration (from wave-drag data).

(b) Configuration with computed intersection.

Figure 3. Example of calculated wing-fuselage intersection.

Figure 4. Example of calculated canard-fuselage intersection.

Figure 5. Intersections of vertical and horizontal tail surfaces with fuselage.

Figure 6. Example of calculated pylon-nacelle intersection.

Figure 7. Intersection of vertical fin with wing upper surface.

Figure 8. Intersection of pylon with wing lower surface.

Figure 9. Illustration of fin input problem that causes algorithm to fail. Extension of each fin lofting line intersects fuselage in two places.

Figure 10. Example of input configuration in wave-drag format.

Figure 11. Example of output configuration with vertical fin on fuselage without canard.

Figure 12. Example of output configuration with vertical fin on wing with canard.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1994	Technical Memorandum

4. TITLE AND SUBTITLE
Automatic Procedures for Computing Complete Configuration Geometry From Individual Component Descriptions

5. FUNDING NUMBERS
WU 509-10-11-01

6. AUTHOR(S)
Raymond L. Barger and Mary S. Adams

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17395

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-4607

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

Subject Category 02

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Procedures are derived for developing a complete airplane surface geometry starting from component descriptions. The procedures involve locating the intersection lines of adjacent components and omitting any regions for which part of one surface lies within the other. The geometry files utilize the wave-drag (Harris) format, and output files are written in Hess format. Two algorithms are used: one, if both intersecting surfaces have airfoil cross sections; the other, if one of the surfaces has circular cross sections. Some sample results in graphical form are included.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Airplane configuration; Airplane geometry; Surface description; Automatic methods	12
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	